CSC3414
CSC3415

Software Development Project A & B

Faculty of Sciences

Introductory/Study Book
2006

Written by

Dr Richard Watson
Department of Mathematics and Computing
The University of Southern Queensland

Published by

The University of Southern Queensland
Toowoomba Qld 4350
Australia

http://www.usq.edu.au/

©The University of Southern Queensland, 2006

Copyrighted materials reproduced herein are used under the provisions of the Copyright Act 1968 as amended, or as a result of application to the copyright owner.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recording or otherwise without prior permission.

This book was set in Computer Modern Roman and Helvetica by the author, using the LaTeX typesetting system and a set of special macro definitions. It was produced on a IBM compatible PC running under RedHat Linux.

Contents

Course Specification iii

1 Course Organisation 2

1.1 The project . . . 2
1.2 Role of the Course Examiner . . . 2
1.3 Role of the Student . . . 2
1.4 Computing Platform . . . 3
1.5 Resources . . . 4
1.5.1 Course web page . . . 4
1.6 Residential School . . . 4
1.7 Communication . . . 4
1.7.1 Linux Installation Problems . . . 4
1.8 Support . . . 5
1.8.1 Types of enquiries . . . 5
1.8.2 Types of support . . . 5

2 Timetable 6

3 Software Engineering Major 7

3.1 Semester 1 tasks . . . 7
3.2 Semester 2 tasks . . . 7
3.3 Record keeping . . . 7
3.4 Evolving documents . . . 8
3.5 Document conventions . . . 9
3.6 User Requirements documents . . . 9
3.7 Submission of project components . . . 9
3.8 Assessment requirements . . . 9

4 Games and Creative Technologies Major 16

4.1 Programming Languages . . . 16
4.2 Project Areas . . . 16
4.2.1 Game Engine Development . . . 16
4.2.2 Game Middleware . . . 16
4.2.3 A Demonstration Game . . . 17
4.3 Assessment requirements . . . 17

References 20

Course Specification

Insert course specifications here

Welcome to the Software Development Project courses. These two courses are intended to be taken in the final year of your program, and are mandatory courses within the Software Engineering major, and the Games and Creative Technologies major, of the Bachelor of Information Technology.

These courses are unlike most other courses that you have undertaken at the university. There is no specific textbook and/or study book to provide technical infromation and there is no examination.

Instead you have the opportunity, and the challenge, to apply knowledge that you have acquired throughout your program in order to complete a large software project.

The project is completed over two semesters. Course CSC3414 is undertaken in Semester 1, and is comprised mostly of analysis and design activities. The Semester 2 course, CSC3415, focuses on implementation and testing.

These courses have many aims. The major ones are:

- To allow you to demonstrate that you are able to apply knowledge gained from previous courses.
- To provide you with the experience of working on a software project of significant size. You will be writing thousands of lines of code. Obviously this will require a high level of planning on you part. It also prepares you for some of the kind of work that you will encounter in the IT industry after you graduate.
- To give you practice in creating design and other project documents.

Details of the procedures and requirements follow. The most important ingredient for success in these courses is your own motivation. Right from the beginning you must work consistently on this project. Planning is vital. Continual progress (even if it seems small) is vital. If you fail to adopt a professional and consistent approach to completing these courses you will find it very difficult to pass. You simply cannot leave important work until the last few weeks of the semester.

You should adopt a professional approach to your work. This is not "just another assignment" which you must submit to get your degree. It is a practice run for your future career. So it behoves you to work like a professional, that is:

- Be well organised.
- Produce high quality work that responds to the (imaginary) client's needs.
- Work consistently and deliver on time.

Please don't read these warnings in a negative light; these courses may well be some of the most rewarding that you have undertaken during your program. When approached with an enthusiastic and confident attitude, these courses can be as exciting and satisfying as they are important for your development as a software engineer.

The project requirements differ between the Software Engineering Major and the Games and Creative Technologies Major. Students studying the Software Engineering Major should complete the requirements specified in section 3. Students studying the Games and Creative Technologies Major should complete the requirements specified in section 4.

1 Course Organisation

1.1 The project

In these courses you are provided with a fairly detailed set of requirements for a software system. Your job is to analyse these requirements, then design, implement and test a system to meet these requirements. Along the way you must demonstrate skills in planning the project and tracking its progress, in creating various analysis and design documents, and in documenting the user interface to your system.

It is vitally important you realise that, from an assessment perspective, we are interested just as much in *how you developed the system* as we are in the final software product. See the assignment specifications in sections 3 or 4 for details of what you must do and what assessment weight is carried by each item.

1.2 Role of the Course Examiner

The course examiner has three basic roles.

1. To act as the 'client'. Imagine that he or she is paying you to do this work. You want to do a good job and produce what the client really wants. So if you have any uncertainty about what you should do based on the (perhaps limited) user requirements document you must contact the examiner to clarify the position.
2. To act as a mentor. If you are unsure how to proceed, or perhaps have some technical questions, do not hesitate to contact the examiner.
3. To assess the completed work.

The examiner will be available for consultation either by telephone, email or face to face at his or her office.

Note that in some cases, the course examiner may nominate someone else to assume these roles and so become your project supervisor. Nonetheless, the examiner has final responsibility in awarding grades for students of the course.

1.3 Role of the Student

Obviously your major role is to finish the job on time, completing all assessable items satisfactorily, and within budget!

However, in order to do this you may well have to undertake some private study of your own in order to learn new skills which you will need to apply in the development of the project. You may also need to re-learn topics that were covered earlier in your studies, or perhaps study some of these topics in more depth than was done originally.

You cannot assume that, at the beginning of these courses, you have all the knowledge and skills necessary to complete it satisfactorily.

There are two other key requirements:

1. **You must communicate regularly** with the course examiner or the supervisor nominated by the examiner. Just as a software project manager expects

and receives regular progress reports from software engineers working on a project, the course examiner expects to hear from you.

Specifically, you **must** report on a weekly basis. This can be done either face to face (for on campus students) or via email (students in either mode). An electronic report submission via the course web page (see section 1.5.1) may be implemented.

The examiner may arrange a weekly group meeting — attendance and verbal reporting at such a meeting would be sufficient.

You will find that maintaining a diary (see section 3.3) will help you greatly in reporting weekly progress to the examiner. Reports can be brief, but they must be detailed. You must be able to report time spent on specific tasks, rather than general comments about your progress.

Note that communication and reporting is assessable; 10% of each assignment's marks are assigned for this component.

2. **You must do your own work.** It is quite acceptable to discuss the design and implementation with other students, and to assist each other in overcoming technical problems, but the actual material that you submit for assessment must be your own work.

   It is completely unacceptable to copy even small parts of another student's work, or for a number of students to collaborate to produce an identical item.

   Unacknowledged copying is plagiarism. The University has strong regulations which provide substantial penalties for students shown to have plagiarised.

A final strong recommendation:

- **Do your best.** Aim to do a really good job. View this project as preparation for your future career. Rather than thinking "What is the minimum I need to do to pass?", ask instead "What can I do to show that I really know my job?".

1.4 Computing Platform

You may implement your project under ether Microsoft Windows (2000 or XP only) or Linux (the version distributed on the Department of Mathematics and Computing CDROM, 2006 set). Other commercial platforms are not suitable as we do not have access to such systems for testing purposes.

The examiner will need to be able to install and run your software. You cannot rely on the availability of proprietary software such as Visual Basic. (Note however that most applications like VB allow the user to build an executable version of the software which does not require the development environment.)

You must ship a final implementation that can be installed and run on a bare Microsoft Windows or Linux system. For example, you cannot assume that Java is installed. So, if you have developed a Java-based system, your software distribution would need to include the Java runtime system as well as your own project software, together with installation instructions for both items.

If you use the Linux platform, you may use any standard application shipped with the Department's current distribution of Linux.

1.5 Resources

It is assumed that most of the resources that you will need to complete this project will come from courses that you have undertaken in the past. As discussed in section 1.3, you may need to access other material to advance your skills and knowledge.

The bibliography at the end of this book lists a selection of possibly useful texts in the areas of software engineering, and systems analysis and design. You should also consult the USQ library.

1.5.1 Course web page

An online learning environment for these courses is available at

`http://www.sci.usq.edu.au/moodle/`

Relevant information will be placed there from time. In particular, and most importantly, the project specification will appear there.

You should check this online learning environment regularly during the duration of the project.

1.6 Residential School

There is no residential school for these courses. If however you will be in Toowoomba attending a residential school, feel free to contact the course examiner if you wish to discuss any aspects of the course.

1.7 Communication

A number of avenues are available for you to communicate with the University and the course examiner.

- You can contact the course examiner via the email facility at the learning environment at

  `http://www.sci.usq.edu.au/moodle/`

  This is the preferred method for enquiries about academic aspects of the courses.

- You can also contact Outreach Services or the International Office, depending on your enrolment status. You can use this contact point for either administrative or academic enquiries. See below (under Student Enquiries) for details. Academic enquiries will be forwarded to the course examiner.

1.7.1 Linux Installation Problems

A systems programmer is employed in The Department of Mathematics and Computing to support staff and students in installing Linux. If you have problems installing Linux, first of all re-read the instructions and try again. If you still have problems contact Linux support via either of the following means:

- email to `linux@usq.edu.au`
- phone the Departmental secretary (07 4631 5555), or the DeC phone number listed in the following section, and ask to be put through to Linux support.

Please, PLEASE, PLEASE, before doing this investigate all options by reading the documentation, and make clear notes about your problem such as the precise steps you followed and any messages you received.

1.8 Support

1.8.1 Types of enquiries

You have access to a wide variety of support services at USQ. Follow the details below or visit the 'Current Students' website at
`http://www.usq.edu.au/currentstudents/default.htm` for more information.

1.8.1.1 General enquiries

USQAssist is the most efficient method for requesting assistance for:

- administrative queries
- assignment submissions
- study assistance
- contacting your lecturer.

1.8.1.2 Technical enquiries

Enquiries relating to access to USQConnect, the USQ*StudyDesk*, or other technical issues can also be directed to USQ*Assist*. If you cannot access USQ*Assist*, contact the Student IT HelpDesk on +61 7 4631 1510 or email `usqconnect@usq.edu.au` for assistance. The Student IT HelpDesk is staffed weekdays between 8.00am and 5.00pm (AEST-Australian Eastern Standard Time), with voicemail after hours.

1.8.2 Types of support

There are a number of ways of accessing support services.

1.8.2.1 Online support

USQ*Assist* is a web self-serve facility for you to:

- find answers to common questions at any time
- ask any question
- track the progress of your question
- keep a record of questions and responses.

To access USQ*Assist* go to `http://usqassist.usq.edu.au` or click on 'USQ*Assist*' in USQ*Connect*.

1.8.2.2 Telephone support

If you prefer to telephone, call Outreach Services on 07 46312285 for assistance. Outreach Services is staffed weekdays between 8.30am and 5.30pm (AEST), with voicemail after hours. If you are located in Eastern Australia, contact your Regional Liaison Officer.

International students telephone the USQ International Office on +61 7 46312362, or your Agent. USQ International is staffed weekdays between 9.00am and 5.00pm (AEST), with voicemail after hours.

1.8.2.3 Fax

International students fax the USQ International Office on +61 7 46362211. All other students fax the Distance and e-Learning Centre on 07 46361049.

1.8.2.4 Postal address

The Administrator Distance and e-Learning Centre University of Southern Queensland Toowoomba Qld 4350 Australia

2 Timetable

These courses should be undertaken over two consecutive semesters. As there is no set programme of study the only timetable elements are the due dates for the assignments. The assignment due dates are as follows:

Course	**Assignment**	**Date Due**
CSC3414	1	14 April 2006
	2	16 June 2006
CSC3415	1	8 September 2006
	2	3 November 2006

3 Software Engineering Major

The full project extends over two semesters. The first semester concentrates on the tasks of system analysis, system design and project planning. No code should be written, except perhaps to prototype some features like user interface or to try out unfamiliar implementation techniques or tools.

The second semester is devoted to implementing the project. In addition to coding the system, a comprehensive test plan must be drawn up and a testing system implemented. User level documentation is created.

There are two assessment items due each semester. The exact details of each assessment item are described in section 3.8. The due dates appear in section 2.

3.1 Semester 1 tasks

1. Analyse the informal user requirements document provided to you and produce a precise and detailed definition of requirements.

2. Design the software system. This ranges from overall architecture to precise details such as definition of class templates, record or database structure, and user interface dialogues.

3. Plan the complete project. This involves identifying tasks, estimating how long they will take to complete, and determining when they will be undertaken. You will also record actual hours spent performing each task.

3.2 Semester 2 tasks

1. Develop and use a comprehensive test plan. The following suggests some test plan activities — it is neither comprehensive nor is it prescriptive. You may add or delete activities as fits your implementation environment. However you must have a clear and comprehensive plan.

    - document the test cases (inputs and expected outputs)
    - plan when to test
    - record test results
    - automate the test process to enable efficient repeated tests

2. Implement the system. This includes developing installation instructions, packaging the software for distribution and possibly providing an installation program or script.

3. Produce user documentation. This is a well structured document describing how to use the system. It is separate from installation instructions.

3.3 Record keeping

Here is a very strong recommendation: **keep a log or diary** of time spent on this project. Each day you should log the hours spent against the task(s) you are working on. This can be done electronically, using a spreadsheet, simple text file, or special purpose project planning software. Also keep a paper copy in a standard diary —

#	Task	Estimate (hours)	Actual hours per week				Total
			week 1	week 2	week 3	···	
1	Blah blah	32	2	7	2	···	11
2	Grumble	50	6	0	1	···	7
3	Hurrumph	18	0	5	6	···	11
	··· ··· ···						
	Total	···	8	12	9	···	29

Table 1: Sample task list showing estimates and progress

these entries are less likely to be lost, and it is very quick and easy to jot down a few lines in a diary, then later log them on the computer.

Keeping a diary is essential if you want to accurately keep track of time expended, and you are required to present these details in the reports.

You should maintain a table of tasks and effort expended against each task. The list of tasks is established early in the project and will most likely grow as the project progresses. You may wish to use a spreadsheet to record this data.

A typical format is depicted in Table 1. You are required to submit multiple versions of this task list, at various times during the project. Later versions differ from earlier ones by having more columns completed, and perhaps by having more rows as more tasks are identified. However, *do not change initial estimates*. You will not be assessed on how well your actual effort matched your estimated effort — rather you should use the figures to help learn about the difficult task of software cost estimating.

The course web page contains some template documents which you may use to report your progress.

3.4 Evolving documents

You will be asked in section 3.8 to submit the same document on more than one occasion. For instance you must provide a task log four times. Also, you are asked to provide a number of versions of the analysis and requirements documents.

When submitting a revised document you must include either as an annotation inside the actual document, or as an accompanying commentary, some indication of what has changed since the previous version of the document was submitted. Changes to the task log are easy to describe (e.g. added new tasks..., new logs for weeks 7...12) but the other documents will require more effort. For requirements documents, one workable and widely used scheme is to include a section (or maybe an appendix) in the document which provides a log of changes made. For example the change log could be a simple chronological list which might in part look like:

2/4/02	added new requirements 3.4, 3.5
22/4/02	modified requirement 7.4 to permit supervisor as well as clerical staff to modify database.
5/5/02	...

3.5 Document conventions

You will produce many documents as part of the assessment process. These should be well organised, readable and well formatted. Hand written documents are not acceptable. All documents should contain the following:

- A title; larger documents should have a separate title page.
- The name of the author.
- The date that the document was produced.
- The version of the document.
- Numbered sections. The use of numbers allows cross referencing within and between documents (a characteristic of good documentation) and makes it easy for the reader to understand the hierarchical nature of the document.

3.6 User Requirements documents

The project specification will be available from the online learning environment

`http://www.sci.usq.edu.au/moodle/`

by the first week of semester 1 2006. If you have difficulties in accessing this material, please contact the course examiner.

3.7 Submission of project components

Hard copies of reports, suitably bound, should be submitted in the usual manner.

Source and executable code, and online documentation, should be submitted on floppy disk or CDROM. You should always package all files into a single archive file using `tar` and `gzip` for Linux or `zip` for Microsoft Windows environments. At least one backup copy of the archive should be provided if submitted on floppy. A file named `README` should appear on the media with comments about what is in the distribution file and how to unpack it.

If submitting executable code, you must provide instructions and possibly automated procedures for installing the application(s). These may appear in the `README` file or be included in a separate file name `INSTALL`.

3.8 Assessment requirements

Two "assignments" are due each semester. Really they would be better described as "project deliverables". Some of the deliverable items are "work in progress" and updated versions are submitted at a later date. Make sure you read section 3.4 for the correct way of handling the creation and maintenance of these evolving documents.

The weighting attributed to each assignment is a per-course contribution. For example, the first assignment for CSC3414 contributes 50% of the marks for course CSC3414.

CSC3414	Software Engineering Major
Assignment 1	weighting 50%

Submit the following items. They should be well formatted and carefully prepared documents. Hand written documents are not acceptable. See section 3.5 for more details about document format.

1. **User requirement analysis** **weight 10%**
   Read and analyse the user requirements presented in section 3.6 and produce a report on your findings. This should be written in natural language but well organised and structured. The report should do *at least* the following:

   - Identify ambiguities and inconsistencies in the report.
   - Identify missing or incomplete information.
   - Describe how you intend to handle these problems. Typically you will either need to
     - make assumptions (which must be fully documented) or
     - seek clarifications from the "user" (which must also be fully documented).

   Include any other comments that you think are important in this report.

   References: See [12] (chapter 4) or [13] (chapter 5) for more on analysing user requirements.

2. **System requirements document** **weight 30%**

   This is a more detailed and complete version of the User Requirements document. The extra detail comes to a large extent as a result of the analysis phase described in the previous item. For instance, an assumption or clarification documented as such in the analysis phase will become a requirement in this current document. Where appropriate, each requirement should be supported by a justification and possibly a cross reference to the requirements analysis document or the user requirements document.

   The system requirements document sets out what you plan to deliver. *It does not describe design or implementation details.*

   The major element of the document is a complete, enumerated, well structured, and detailed list of requirements.

   If necessary, the document may be supported by models showing for example

   - relationships between elements of the user domain,
   - flow of information in the system,
   - state transitions, and
   - timing of events.

   References: See [12] (chapters 4 and 7) or [13] (chapter 5) for more on defining systems requirements.

3. **Preliminary plan and activity log** **weight 5%**

- Develop a preliminary list of tasks required to complete the project. This includes tasks undertaken in both of the semesters.
- Estimate the amount of time required to undertake each task. You should use hours as a measure of duration, bearing in mind that you should be spending about 10 hours/week on these courses.
- Record the time (in hours) spent undertaking each activity.
- Present the results in a form similar to that shown in table 1. Note that you may wish to break major tasks into subtasks.

4. **Communication** **weight 5%**

   As described in section 1.3, communication with the examiner or nominated supervisor is vitally important. You will be assessed on the regularity, timeliness, accuracy and relevance of your interactions.

CSC3414	**Software Engineering Major**
Assignment 2	**weighting 50%**

Submit the following items. They should be well formatted and carefully prepared documents. Hand written documents are not acceptable. See section 3.5 for more details about document format.

1. **Updated User requirements documents** **weight 5%**

   This is an updated version of the documents 1 (User requirements analysis) and 2 (System requirements) produced in Assignment 1 earlier in the semester. Make sure that the changes to these documents are clearly described as discussed in section 3.4.

2. **Software Design Document** **weight 35%**

   Produce a document which describes *in great detail* what you are going to produce. You should be able to hand *only* this document to another programmer who knows nothing of the user requirements and expect the correct product to be produced.

   The following is a suggested list of some of the items that should be contained in the document. You will almost certainly need more than these. You will almost certainly not need to include some of these.

   - System architecture.
   - Detail data structure design. Examples are file or database formats.
   - Class templates in the target language.
   - Structure charts.
   - Data flow diagrams.
   - Pseudo code.
   - State transition diagrams.
   - User interface specifications.
     - How many screens? What are the input fields on each screen? What validation checks are made on fields? What are the initial values for fields?

– If command line, what is the syntax for commands?

3. **Updated plan and activity log** **weight 5%**

   This is the latest version of the same item submitted in the first assignment. It should reflect *all* the work done during the semester.

4. **Communication** **weight 5%**

   As described in section 1.3, communication with the examiner or nominated supervisor is vitally important. You will be assessed on the regularity, timeliness, accuracy and relevance of your interactions.

CSC3415	Software Engineering Major
Assignment 1	weighting 40%

In the course CSC3415 you will implement and test the system designed in course CSC3414. The first assessment focuses on testing. Of course you will be occupied mostly with coding, but it is imperative that you begin testing early. Hence we demand that you produce a comprehensive test plan. You should also automate the testing process as much as possible. This may mean writing extra scripts or programs which exercise your code and record the results. This will enable you to regularly test your code. (Note that automated testing with GUI application development systems like Visual Basic is difficult, and will not be expected.)

Submit the following items. They should be well formatted and carefully prepared documents. Hand written documents are not acceptable. See section 3.5 for more details about document format.

1. **Updated requirements documentation** **weight 5%**

   During implementation you will no doubt discover that changes are required to your original design. Edit the following documents, and include a change log, to reflect these changes.
   - User requirements analysis
   - System requirements
   - Software design

2. **Test plan** **weight 20%**

   Describe your approach to testing. For example (this is not an exhaustive list):
   - How do you test?
   - When do you test?
   - Have you created special test programs and scripts?
   - How do you record test results?

   You must document the test cases; each test case describes the required result from a given set of inputs and a specified system state.

   See [12] (chapters 22 and 23) or [13] (chapters 19 and 20) for more on testing.

3. **Version 0 system** **weight 5%**

   By this time you should have some executable code. It may be just the top level with some stubs, or some bottom level elements with some test driver programs.

   You should submit this code together with installation instructions. We require this for a number of reasons:
   - To gauge that you are making progress.
   - To ensure that the system will in fact install on our software platform.
   - To ensure that you think about the installation process now, rather than doing it in a rushed (and possibly inadequate) manner at the end of the project.

4. **Updated plan and activity log** **weight 5%**

This is the latest version of the same item submitted in the previous semester. It should reflect *all* the work done during last semester and this semester.

5. **Communication** **weight 5%**

   As described in section 1.3, communication with the examiner or nominated supervisor is vitally important. You will be assessed on the regularity, timeliness, accuracy and relevance of your interactions.

CSC3415	**Software Engineering Major**
Assignment 2	**weighting 60%**

This is the final submission. Make especially sure that everything is nicely packaged, well organised and neatly formatted. See section 3.5 for more details about document format. This is what is required:

1. **Final version of requirements documentation** **weight 3%**

   Submit the following documents.
   - User requirements analysis
   - System requirements
   - Software design

2. **Final version of plan and activity log** **weight 4%**

   This should reflect *all* the work done during both semesters of the project.

3. **User documentation** **weight 10%**

   This is an instruction manual for the user of your system. It does not contain installation instructions. Assume the user knows nothing about the technical details of the system. You can assume that the user has reasonable domain knowledge, and will be familiar with the kinds of 'things' with which your system deals.

4. **Test log & updated test plan** **weight 3%**

   Resubmit the final version of your test plan from the previous assignment. Most importantly, include a log showing when testing was carried out and the results of the testing.

5. **The completed executable system** **weight 35%**

   The executable programs that make up the system must be submitted, as well as the source code. You must also include:
   - Instructions to install the system.
   - (Preferable but not mandatory) An installation script or program.
   - A report on system functionality. Students regularly complete more or less features than are required by the specification. You should highlight if and to what extent your system differes from the specified system.
   - A list of known errors. That is, where a system function has been inplemented but does not work correctly, or produces a run time error.

The following aspects of system will be assessed:

- The installation process should be well documented, simple, and error free.
- All functionality defined in the User Requirements document should be met.
- The system should be free of defects — that is it should not crash or behave incorrectly.
- It should be usable — the user interface should be simple and natural to use.

6. **Communication** **weight 5%**

   As described in section 1.3, communication with the examiner or nominated supervisor is vitally important. You will be assessed on the regularity, timeliness, accuracy and relevance of your interactions.

4 Games and Creative Technologies Major

The project for this strand extends over two semesters. Your objective is to develop a major piece of software relevant to computer games. This might be a game engine, game middleware or a game. You can choose to work in a team with other students or on your own. Your project idea must be approved by the examiner in charge of this project before you proceed.

During the first semester you will concentrate your efforts on field research and the design document. You should start writing code, however it will focus on your chosen technologies/programming languages/game engine and not necessarily reflect the final product.

The second semester should be devoted to the development of the final product concluding with a presentation of your work at the end.

4.1 Programming Languages

The programming language of choice in the games industry is C/C++. While you are encouraged to use C/C++ note that there is a growing area using Java. Therefore you can select C/C++ or Java to work on your software.

4.2 Project Areas

It is envisaged your project will fall into one of the following 3 areas. Once you have decided on one, please follow the advice given:

4.2.1 Game Engine Development

The objective of developing a game engine would be to focus on the computer graphics and sound elements of a game. A game engine is a piece of software that takes care of all the routine and difficult game programming code. For example, Apocalyx 3D (see http://apocalyx.sourceforge.net/) is a game engine written in OpenGL and C++. You might decide to create a compact version of this engine or write one in Java using Apocalyx as a guide. Alternatively, you might decide to create an engine dedicated to creating strategy games. To get started in creating a game engine see OpenGL Game Programming by Hawkins & Astle for a chapter dedicated to game engine creation.

Your final product must include a game written with the engine.

4.2.2 Game Middleware

Middleware is software that sits between the game engine and the game code. It usually concentrates on elements such as Artificial Intelligence, Physics, Sound and many others. The idea of middleware is to empower the game creators with libraries that do all the hard work in relation to functionality not provided in game engines. One example might be an Emotion Engine that generates emotions in game characters. Another might be a Physics Engine for racing game that concentrates on all the physics elements of driving a car at different speeds on different road surfaces.

If you decide to write a piece of middleware, you will need a game engine to sit it on. You can use the engine developed in OpenGL Game Programming by Hawkins & Astle, the Apocalyx 3D game engine (see http://apocalyx.sourceforge.net/) or any other you can obtain. A search on the Internet will reveal many open source game

engines.

4.2.3 A Demonstration Game

The third alternative project for this course is to create a fully functioning game demo. Obviously since games today take millions of dollars and huge teams of people to create them, this project will not be comparable to the best selling games on the market. However, if you remove the flashy graphics and audio, you are left with the game play itself. If you have a strong game foundation then it should be fun to play regardless of the graphics. Take for example the original SimCity or Monkey Island.

Your objective should be to come up with an original game concept and to create the game by programming it from scratch or on top of an existing game engine.

No matter which of these types of projects you choose you should be able to sell the ideas to a potential publisher/user. With this in mind, you should read the International Game Developers' Associations Submission Guide (available on the course website). This guide will help you decide which parts of the project need the most emphasis work. In the end, you may not have a fully functioning product, but enough of it needs to be in place that you can show off your abilities and hopefully attract money from publishers to finish the job.

4.3 Assessment requirements

Two "assignments" are due each semester. Really they would be better described as "project deliverables". Some of the deliverable items are "work in progress" and updated versions are submitted at a later date.

The weighting attributed to each assignment is a per-course contribution. For example, the first assignment for CSC3414 contributes 50% of the marks for course CSC3414.

CSC3414	**Games and Creative Technologies Major**
Assignment 1	**weighting 50%**

The following items should be submitted electronically via the course website.

1. **Project Proposal** **weight 10%**

   The project proposal is a 3 - 4 page document outlining your choice of project. You should include the following sections:

   - Definition of the Project
     - What are you going to create?
     - Why have you chosen this as your project?
   - Review of Existing Technologies.
     This section should examine the technologies you have researched in coming to a conclusion about how to proceed. You should address questions such as:
     - What game engines have you examined? How do they work? What games have been written with them? Are they open source?

  - What middleware exists? How has it been written?
  - What types of games already exist that are similar to the game you are proposing?
  - What programming language/development environment are you going to use?
- Team Members
  If this is a team project - who is in the project. What will their role be?
- Timeline
  Create a timeline showing the entire scope of the project and the milestones you will adhere to.

2. **Draft Design Document** **weight 80%**

   - The design document template can be found on the course website. For this assignment you are to submit a draft of the document. The document is an evolving report that you will continue to change throughout the project. Each time the document is changed the person who changed it and what they changed should be documented in the first few pages.
   - For this assignment you should complete the Game/Project Overview and Feature set sections of the document.

3. **Log and Communication** **weight 10%**

   - Throughout the project you are required to meet with a supervisor who will monitor your progress and mark your work. You will be assessed on the regularity and relevance of your interactions with the supervisor and any team members.
   - You should also keep a log that should be submitted as the appendix to the design document that records what tasks each person has done, when they did them and how long it took them. Include dates when you have met your supervisor or team members, research time, design time etc.

CSC3414	**Games and Creative Technologies Major**
Assignment 2	**weighting 50%**

The following items should be submitted electronically via the course website.

1. **The completed design documentation** **weight 90%**
2. **Log and Communication** **weight 10%**

CSC3415	Games and Creative Technologies Major
Assignment 1	**weighting 50%**

1. **A preliminary run-time version of your project** **weight 70%**
2. **A preliminary user manual** **weight 20%**

   This should present an outline of the features you have working and how this version can be used.
3. **Log and Communication** **weight 10%**

CSC3415	Games and Creative Technologies Major
Assignment 2	**weighting 50%**

1. **The final run-time version of your project** **weight 70%**
2. **User Manual** **weight 20%**
3. **Presentation/Pitch** **weight 10%**

   Internal students should prepare a 20min presentation of their project which will be assessed by their peers and academic staff. The presentation should be aimed at potential publishers/users of your product/game.

   External students should prepare a PowerPoint slide or Multimedia presentation to best sell their product in lieu of not being able to pitch it in person.

   Please see the documents on the course website for preparing a pitch document.

NOTE:
The project application MUST run without any extra setup and all files must reside in the same directory. Before submitting test this by copying the executable and data files onto a CDROM and try running it on another computer from the CDROM. If you submit your product to a game publisher - it will need to run first time or they will throw the CDROM in the bin!

References

[1] A. Davidson. *Killer Game Programming in Java*. O'Reilly & Associates, 2005.

[2] Penny Baillie de Byl. *Programming Believable Characters for Computer Games*. Charles River Media, 2004.

[3] Alan Dix, Janet Finlay, Gregory Aboud, and Russell Beale. *Human-Computer Interaction*. Prentice Hall, London, 2nd edition, 1998.

[4] M. Fowler and K Scott. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, 2nd edition, 2000.

[5] K. Hawkins and D. Astle. *OpenGL Game Programming*. Premier Press, 2004.

[6] I. Jacobson. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.

[7] K.E. Kendall and J.E. Kendall. *Systems analysis and design*. Prentice Hall, 4th edition, 1998.

[8] Leszek Maciaszek. *Requirements analysis and system design*. Addison-Wesley, 2001.

[9] M. McShaffry. *Game Programming Complete*. Paraglyph Press, second edition, 2005.

[10] R.S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 3rd edition, 1992.

[11] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modelling and Design*. Prentice Hall, 1991.

[12] Ian Sommerville. *Software Engineering*. Addison-Wesley, 5th edition, 1995.

[13] Ian Sommerville. *Software Engineering*. Addison-Wesley, 6th edition, 2000.